SPE-223907-MS

Coupled Graph Neural Network and Fourier Neural Operator Architecture for Ensemble Workflows in 3d Reservoir Simulation

Surya Teja Sathujoda and Soham Sheth, SLB

Abstract

Current reservoir simulators solve Advection-Diffusion-Reaction equations on unstructured grids to a high degree of accuracy but are computationally expensive to use in history matching, optimization and uncertainty quantification workflows. We propose a novel machine learning architecture to auto-regressively predict state variables and well outputs of a reservoir for an ensemble of cases with varying well configurations. This new approach constructs a Graph Neural Network (GNN) model to predict hyperbolic variables (i.e. saturations and compositions), a Fourier Neural Operator (FNO) model to predict elliptic variables (i.e. pressure), and a feed forward layer for well outputs. Predictions from the FNO model are fed into the GNN and are coupled in an auto-regressive manner to predict an arbitrary future state of the system given just the initial state of the reservoir. The outputs of the model at each timestep are then fed into the well output layer to predict outputs for each injector and producer well in the reservoir. Results are compared with solutions from a high-fidelity reservoir simulator applied to an ensemble workflow.

Introduction

Reservoir engineering workflows such as history matching, uncertainty quantification and optimization require vast amounts of compute power to simulate the necessary realizations in a given ensemble. Solving the inverse problem of finding the optimal controls to maximize production rates of wells for instance, would require a reservoir simulator to be run hundreds, if not thousands of times to converge to an acceptable solution. It is hence highly desirable to develop a fast proxy model which accurately predicts the solutions of the reservoir simulator at only a fraction of the time. Machine Learning models have become a prime candidate for this proxy modelling behavior as, once trained, they have been shown to accurately perform inference several orders of magnitude faster than state-of-the-art reservoir simulators. Some of the challenges that machine learning proxy models need to address before they can rival traditional simulators however is that they need to:

- Perform inference on unstructured grids
- Generalize to unseen geological properties (porosity, permeability, etc…)

- Handle varying well controls
- Handle new well locations
- Handle scaling to large grids
- Handle variable timestep roll outs
- Exhibit resolution independence
- Exhibit mesh-independence

Attempts to tackle these challenges in previous works are described below.

Early on, some general machine learning models were first applied to this problem by treating the initial field values at each point as the input to the model and the numerical solution to the PDE as the output for time-independent cases and the next step field values as the output for time dependent cases. These were purely data-driven approaches which were agnostic to the form or the physics of the underlying PDEs governing the system. Some of these early methods, which now serve as a benchmark for which to compare more advanced methods to, are: **RBM**: the classical Reduced Basis Method (using POD basis). **FCN**: a state-of-the-art neural network architecture based on Fully Convolution Networks (Zhu Y 2018). **PCANN**: an operator method using PCA as an auto-encoder on both the input and output data and interpolating the latent spaces with a neural network (Bhattacharya, et al. 2021). For time dependent tasks, popular models in other time dependent regression tasks were applied such as: **RESNET**: a residual learning framework to ease the training of networks that are substantially deeper than those used previously (He K 2016). **U-NET:** A popular choice for image-to-image regression tasks consisting of four blocks with 2-d convolutions and deconvolutions (Ronneberger O 2015).

As developments continued, one of the main directions of success to model non-linear control dynamics came from the Embed-to-Control **E2C** model (Watter M 2015). The E2C model consists of a variational auto-encoder and a transitional block which learns to generate image trajectories from a latent space in which the dynamics are constrained to be locally linear. This work was subsequently modified by replacing the variational auto-encoder with an ordinary encoder-decoder structure and then applied to the problem of 2D reservoir surrogate modelling by (Jin ZL 2020). The results produced by this model were extremely accurate compared to previous methods and formed the bed rock for the expansion of the model to predict well outputs and for resource extraction in (Coutinho EJR 2021). The thus proposed Embed to Control and Observe **E2CO** model added an additional parameterized network flow in the transitional block to predict well outputs such as flow rates at source/sink points. Both the E2C and the E2CO models were naturally expanded to 3D grids by (Atadeger A 2022) by re-modelling the architecture to involve 3D convolution blocks and transposes in the encoder and decoder respectively to more reflect real-world systems in the area. Then a localized-learning method was proposed to overcome the drawback of large training times of convolution methods in (Sathujoda S 2023).

Recently, a new branch of proxy modelling architectures has come to the forefront. That is the application of Fourier Neural Operators for this time-dependent problem. Some of the key works in this area are (L. Z. Wen G 2022), (Zhang K 2022), (L. Z. Wen G 2023), (Jiang Z 2024). Fourier Neural Operators are described in detail in the following section.

Finally, to tackle the problem specifically of hyperbolic variables such as saturation and compositions, Graph Neural Networks has shown increased accuracy. Some of the key works in applying GNNs to reservoir simulation are (Sathujoda S. 2024), (Sasal, Busby and Hadid 2024), (Huang, Gong and Sun 2023). The formulation of GNNs is in the following section.

In this work, we develop a coupled workflow of Fourier Neural Operators (FNOs) (Li Z 2021) and Graph Neural Networks (GNNs) (Thomas N. Kipf 2017) along with a dedicated feedforward neural network to auto-regressively predict state variables and well outputs over time for a large 3D reservoir with varying well controls. As shown in (L. Z. Wen G 2022), FNOs perform with a high degree of accuracy when approximating solutions to elliptic partial differential equations (such as the pressure equation) and scale well in training efficiency when compared to convolution-based approaches and graph-based approaches. GNNs, while slower to train than FNOs, perform inherently well in predicting solutions to hyperbolic partial differential equations (such as the saturation equation) (Sathujoda S. 2024) due to their local update procedure. They also have an inductive bias towards unstructured grids and have the advantage of better generalization to unseen geologies and varying well controls as the rate of information propagation in the graph can be heuristically controlled using edge attributes. We hence aim to take advantage of both the FNO's and GNN's characteristics to develop a coupled architecture to accurately predict pressures and saturations simultaneously for varying well controls.

In mathematical background section, we describe the theoretical formulation of Fourier Neural Operators and the Graph Neural Networks. In the methodology section, we describe the reservoir model which we aim to model and detail the generation of the ensemble used to train and test. We also give a detailed description of the coupled workflow proposed with a well output network and describe details of implementation, such as loss functions and additional technical details used to improve performance. In the results section, we show the results of a 2D case for GNNs and the full 3D case predicted pressures, saturations and well outputs and explain the effectiveness of the method. Finally, in the conclusion we summarize our findings and propose future directions of work.

Mathematical Background

The governing equation of subsurface flow, derived from the law of mass-conservation and Darcy's law, relates the fluid flow and flow potential gradients of a multi-phase porous-medium as given below

$$\Delta \cdot \left[\alpha k \frac{k_{r,m}(S_m)}{\mu_m B_m} (\nabla p_m - \gamma_m \nabla_z) \right] - \beta \frac{\partial}{\partial t}\left(\frac{\phi S_m}{B_m} \right) + \sum_w \frac{q^w_{sc,m}}{V_b} = 0$$

where **k** denotes the permeability tensor, k_r the relative permeability, μ the viscosity, B the formation volume factor, p the pressure, S the saturation, ϕ the porosity, t the time, γ the specific weight, q the source/sink terms, z the depth and V_b the bulk volume. The subscript sc represents standard conditions, m the medium, and α and β are unit field constants.

Fourier Neural Operators

In this section we mathematically define the problem of PDE solution approximation and formalize the framework of Neural Operators and more specifically Fourier Neural Operators. We follow the notations and conventions introduced in (Nikola Kovachki 2023).

A generic parametric PDE can we defined in the following manner,

$$\begin{aligned} (\mathcal{L}_a u)(x) &= f(x), \quad x \in D, \\ u(x) &= 0, \quad x \in \partial D, \end{aligned} \tag{1}$$

where $a \in A, f \in U^*$ and $D \subset \mathbb{R}^d$ is a bounded domain. We assume that the solution $u : D \to \mathbb{R}$ lives in the Banach space U and $\mathcal{L}_a: A \to \mathcal{L}(U;U^*)$ is a mapping from the parameter Banach space A to the space of (possibly unbounded) linear operators mapping U to it's dual U^*. A natural operator which arises from this PDE is $\mathcal{G}^\dagger := \mathcal{L}_a^{-1} f \quad : \mathcal{A} \to \mathcal{U}$ defined to map the parameter to the solution $a \mapsto u$.

Any machine learning approximator of PDE solutions aims to build an approximation of $G^\dagger$ by constructing a parametric map

$$\mathcal{G}_\theta : \mathcal{A} \to \mathcal{U}, \quad \theta \in \mathbb{R}^p \tag{2}$$

with parameters from the finite-dimensional space $\mathbb{R}^p$ and then choosing $\theta^\dagger \in \mathbb{R}^p$ such that $G_{\theta^\dagger} \simeq G^\dagger$.

The proposed architecture for the Neural Operator G_θ: $A \rightarrow U$ can be broken down into three parts. First, a mapping from the initial function space to a hidden representation, then an iterative update of the hidden representation, and finally a projection for the last hidden state back to the output function space. The three steps are respectively called **Lifting**, **Iterative Kernel Integration**, and **Projection** and are mathematically defined by the following

$$\mathcal{G}_\theta := \mathcal{Q} \circ \sigma_T\left(W_{T-1} + \mathcal{K}_{T-1} + b_{T-1}\right) \circ \ldots \circ \sigma_1\left(W_0 + \mathcal{K}_0 + b_0\right) \circ \mathcal{P} \tag{3}$$

where $\mathcal{P} : \mathbb{R}^{d_a} \rightarrow \mathbb{R}^{d_{v_0}}$ and $\mathcal{Q} \quad : \mathbb{R}^{v_T} \rightarrow \mathbb{R}^{d_u}$ are the pointwise fully local lifting and projection mappings respectively. The steps in between are the Iterative Kernel Integration steps defined by the sum of local linear operators $W_t \in \mathbb{R}^{d_{v_{t+1}} \times d_{v_t}}$, Integral Kernel Operators $\mathcal{K}_t \quad : \left\{v_T \quad : D_t \rightarrow \mathbb{R}^{d_{v_t}}\right\} \rightarrow \left\{v_{t+1} \quad : D_{t+1} \rightarrow \mathbb{R}^{d_{v_{t+1}}}\right\}$, bias functions $b_t \quad : D_{t+1} \rightarrow \mathbb{R}^{d_{v_{t+1}}}$ and fixed activation functions σ_t, which act as pointwise local maps $\mathbb{R}^{v_{t+1}} \rightarrow \mathbb{R}^{v_{t+1}}$ in each layer.

The role of the lifting operation is to transform the input to a hidden representation in which it is more efficient for the integral kernel operator to capture non-local interactions of $G^\dagger$. The projection operator simply transforms the hidden representation back to the output space. These operations are usually parameterized by shallow neural networks. The choice of the Kernel Integral Operator K (defined below) delineates the class of the Neural Operator.

Definition 1. (Iterative updates) *Define the update to the representation* $v_t \rightarrow v_{t+1}$ *by*

$$v_{t+1}x := \sigma\left(W v_t(x) + \left(\mathcal{K}(a;\ \phi)v_t\right)(x)\right), \quad \forall\, x \in D \tag{4}$$

where $\mathcal{K} \quad : A \times \Theta_{\mathcal{K}} \rightarrow L\left(\mathcal{U}\left(D; R^{d_v}\right), \mathcal{U}\left(D; R^{d_v}\right)\right)$ *maps to bounded linear operators on* $\mathcal{U}\left(D; \mathbb{R}^{d_v}\right)$ *and is parameterized by* $\phi \in \Theta_K$, $W \quad : \mathbb{R}^{d_v} \rightarrow \mathbb{R}^{d_v}$ *is a linear transformation, and* $\sigma: \mathbb{R} \rightarrow \mathbb{R}$ *is a non-linear activation function whose action is defined component-wise.*

Neural Operators take $K(a;\phi)$ to be a kernel integral transformation parameterized by a neural network.

Definition 2. (Kernel Integral Operator) K *Define the integral kernel operator mapping in (4) by*

$$\left(\mathcal{K}(a;\ \phi\ v_t)\right)(x) := \int_D k(x, y, a(x), a(y); \phi) v_t(y) \mathrm{d}y, \quad \forall\, x \in D \tag{5}$$

where $\kappa_\phi \quad : \mathbb{R}^{2(d+d_a)} \rightarrow \mathbb{R}^{d_v \times d_v}$ *is a neural network parameterized by* $\phi \in \Theta_K$.

Here, κ_ϕ plays the role of a kernel function which we learn from data. A natural selection for the kernel function in machine learning problems is the convolution operator. Inspired by this, the Fourier Neural Operator uses the Kernel Integral Operator defined by Equation (6) below.

Definition 3. (Fourier Integral Operator) K *Define the Fourier integral operator*

$$\left(\mathcal{K}_t(v_t)\right)(x) = F^{-1}\left(R_\phi \cdot F\left(v_{t-1}\right)\right)(x), \quad \forall\, x \in D \tag{6}$$

where R_ϕ *is the Fourier transform of a periodic function* $\kappa\ : \overline{D} \rightarrow \mathbb{R}^{d_v \times d_v}$ *parameterized by* $\phi \in \Theta_K$. *F and* F^{-1} *correspond to the Fourier and Inverse Fourier Transforms*.

This object R_ϕ is parameterized by a linear transformation of the top k modes pertaining to the given layer, which acts as a hyperparameter in the model.

We have now defined all the operators in Equation (3) for the Fourier Neural Operator. A full illustration of the architecture is given in Figure 1.

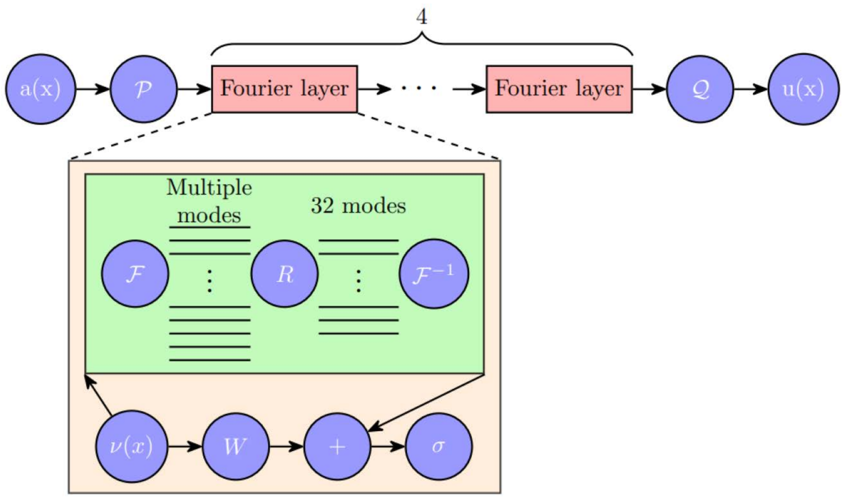


Figure 1—Architecture of the Fourier Neural Operator: The model takes as input a(x) which undergoes a lifting operation, denoted as P. This is followed by n consecutive Fourier layers. Subsequently, a projector Q transforms the data to the desired target dimension, resulting in the output u(x). The inset provides a detailed view of the structure of a Fourier layer. Data initially flows to the layer as v(x) and is bifurcated into two branches: one undergoes a linear transformation W, and the other first experiences a Fourier transformation, from which the k lowest Fourier modes are kept and the other higher modes are filtered out by undergoing a transformation R, and ends with an inverse Fourier transformation with these left modes. The two data streams then converge, followed by the application of an activation function σ. This representation is taken from (Surya Sathujoda 2023) with 4 fourier layers and cut of 32 modes.

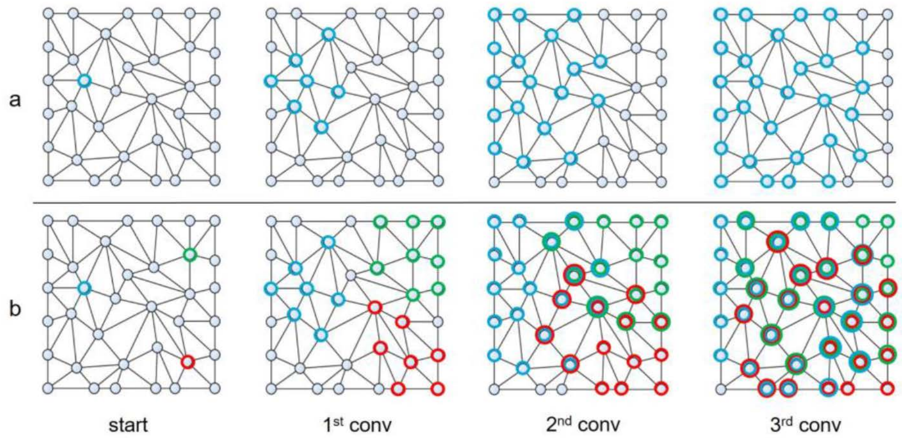


Figure 2—Information propogration of a) a single node b) three nodes over multiple graph convolutions. b) Shows the range of nodes which influence the representation of a single node over several graph convolutions.

Graph Neural Networks

In this section, we mathematically define Graph Neural Networks and specifically define Graph Convolution Networks and the Graph Convolution II Network. Graph Neural Networks are a natural choice of formulation for the reservoir simulation problem as the reservoir can be viewed as a graph with nodes corresponding to the cells and edges corresponding to cell connections.

We follow the formulation from (Sathujoda S. 2024). Let $G=(V,E)$ be a graph with nodes $u \in V$ and edges $e_{uv} \in E$ which represent connections between nodes u and v. Then $N_u=\{v: e_{uv} \in E\}$ is defined to be the neighborhood of node u. Additionally, let $x_u \in \mathbb{R}^n$ represent features associated with node u and $e_{uv} \in \mathbb{R}^m$ the edge features associated with edge e_{uv}. Then we define a Message-Passing Neural Network (MPNN) layer as follows (Michael M. Bronstein 2021):

$$h_u = \phi\left(x_u, \bigoplus_{v \in N_u} \psi(x_u, x_v, e_{uv})\right) \tag{7}$$

where ϕ and ψ and differentiable functions (such as neural networks) and $\oplus$ is a permutation invariant aggregation operator which can take an arbitrary number of inputs, such as element-wise sum or mean. Here, h_u is the updated representation of the features at u. Graph Neural Networks are defined as l MPNN layers applied sequentially to the initial features of the nodes.

Some major GNN architectures that have arisen in recent times include the Graph Convolution Network (GCN) (Thomas N. Kipf 2017), which proposes a localized graph convolution operation that convolves only over direct neighbors of a node normalized by node degrees, and the Graph Attention Network (GAT) (Petar Veličković 2018), which incorporates an attention mechanism to dynamically learn edge weights for the message-passing layer. A Graph Convolution Network is defined by the MPNN layer below,

$$h_u^{(l+1)} = \sigma\left(W^{(l)} \sum_{v \in \mathcal{N}_u} \frac{h_v^{(l)}}{\sqrt{|\mathcal{N}_u| \cdot |\mathcal{N}_v|}}\right) \tag{8}$$

where σ is an activation function (such as ReLU) and $W^{(l)}$ are the weights corresponding to the (shallow) neural network at the lth layer. This can be rewritten in matrix form as following

$$H = \sigma\left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X\Theta\right) \tag{9}$$

where $\tilde{A}$ is the adjacency matrix, $\widetilde{D}$ is the degree matrix, X are the node features and Θ are the neural network parameters.

One of the key advantages of GCNs over other variants of GNNs, such as GATs, is that its message-passing layer is computationally lightweight and training times scale significantly better to larger graphs. This is because these models have less parameters and do not requiring additional attention mechanism computations. GCNs still suffer from the problem of over-smoothing (Qimai Li 2018) however, the phenomenon where the representations of different graph nodes become indistinguishable as the number of message-passing layers increase. To overcome this bottleneck, residual connections and identity mappings were introduced between the layers to maintain a higher signal-to-noise ratio. This architecture, name the Graph Convolution Network II (GCN2) architecture, is given below

$$h_u^{(l+1)} = \sigma\left(\left(\left(1-\alpha^{(l)}\right) \sum_{v \in \mathcal{N}_u \cup \{u\}} \frac{h_v^{(l)}}{\sqrt{|\mathcal{N}_u| \cdot |\mathcal{N}_v|}}\right) + \alpha^{(l)} h_u^{(0)}\right)\left(\left(1-\beta^{(l)}\right) I + \beta^{(l)} W^{(l)}\right) \tag{10}$$

where α is the residual strength and β the identity strength.

Methodology

Reservoir Model

In this work, we use a modified version of the opensource COSTA model (J. Costa Gomes 2023) for machine learning benchmarking. The model is downscaled to 346,920 active cells with 20 vertical injector wells and 22 productor wells. This reservoir is then interpolated onto a (91×95×62) bounding box grid. Representative permeabilities in the x-direction for various layers are shown in Figure 3. The simulation is run in one-year timesteps with injection rates for each well sampled from a Gaussian distribution with mean 4000 STB/day and standard deviation 500 STB/day. Pressure and saturation values of the reservoir for a given timestep and layer are shown in Figure 3.

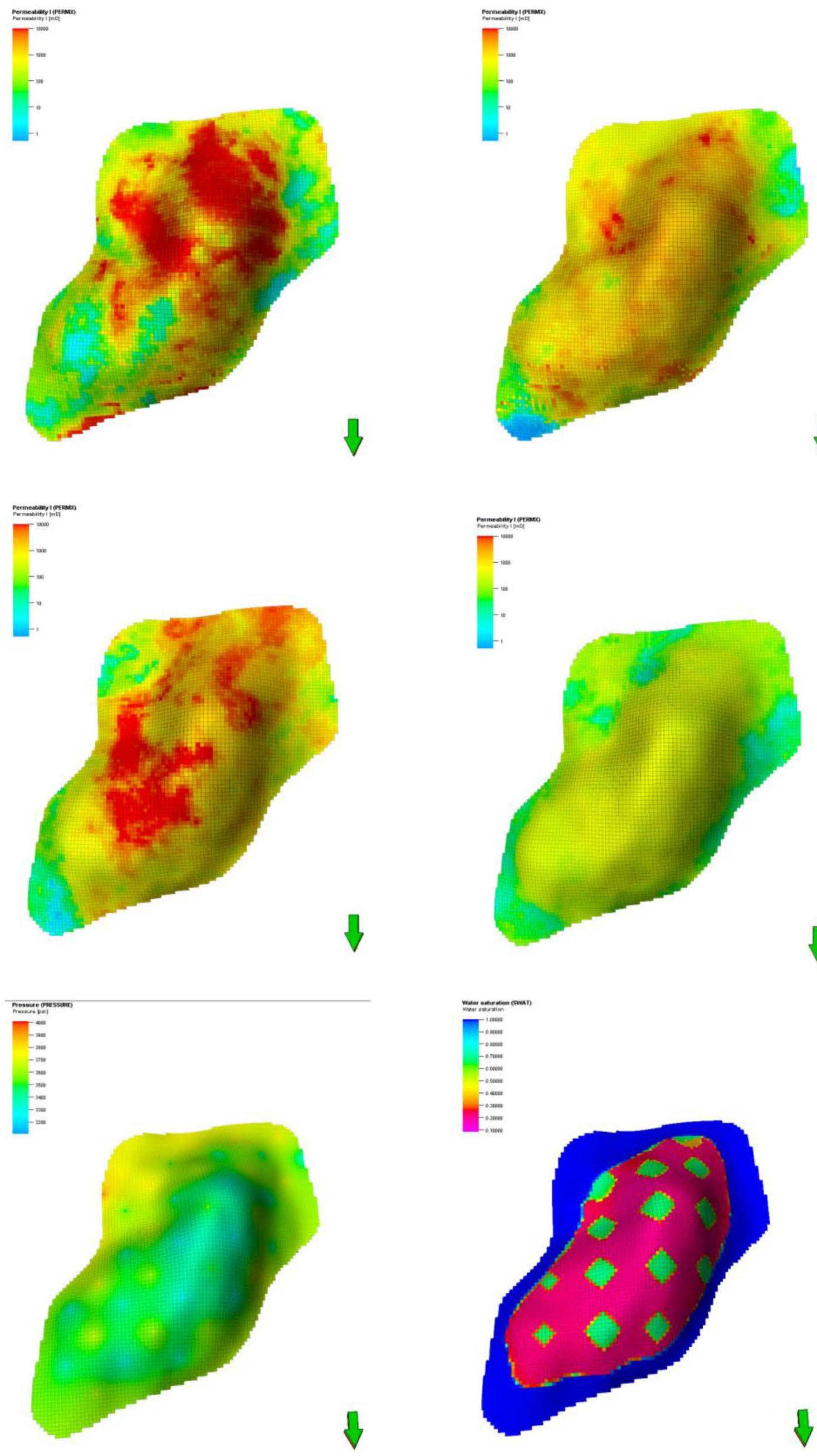


Figure 3—From Left to Right and Top to Bottom: PERMX at z=5; PERMX at z=10; PERMX at z=20; PERMX at z=40; Pressure at z=40, timestep=2036; Saturation of Water at z=40, timestep=2036.

To create the dataset for the machine learning workflow, we create an ensemble of 20 cases with varying well controls and simulate for 16 years. We extract the pressures, saturations, permeabilities in the x, y, and z directions and pore volume for the cells. We also generate the injection rates and bottom hole pressures for the injector wells, and the bottom hole pressures, production rate of oil and production rate of water for the producer wells. A subset of the injection rates in the ensemble are given in Figure 4.

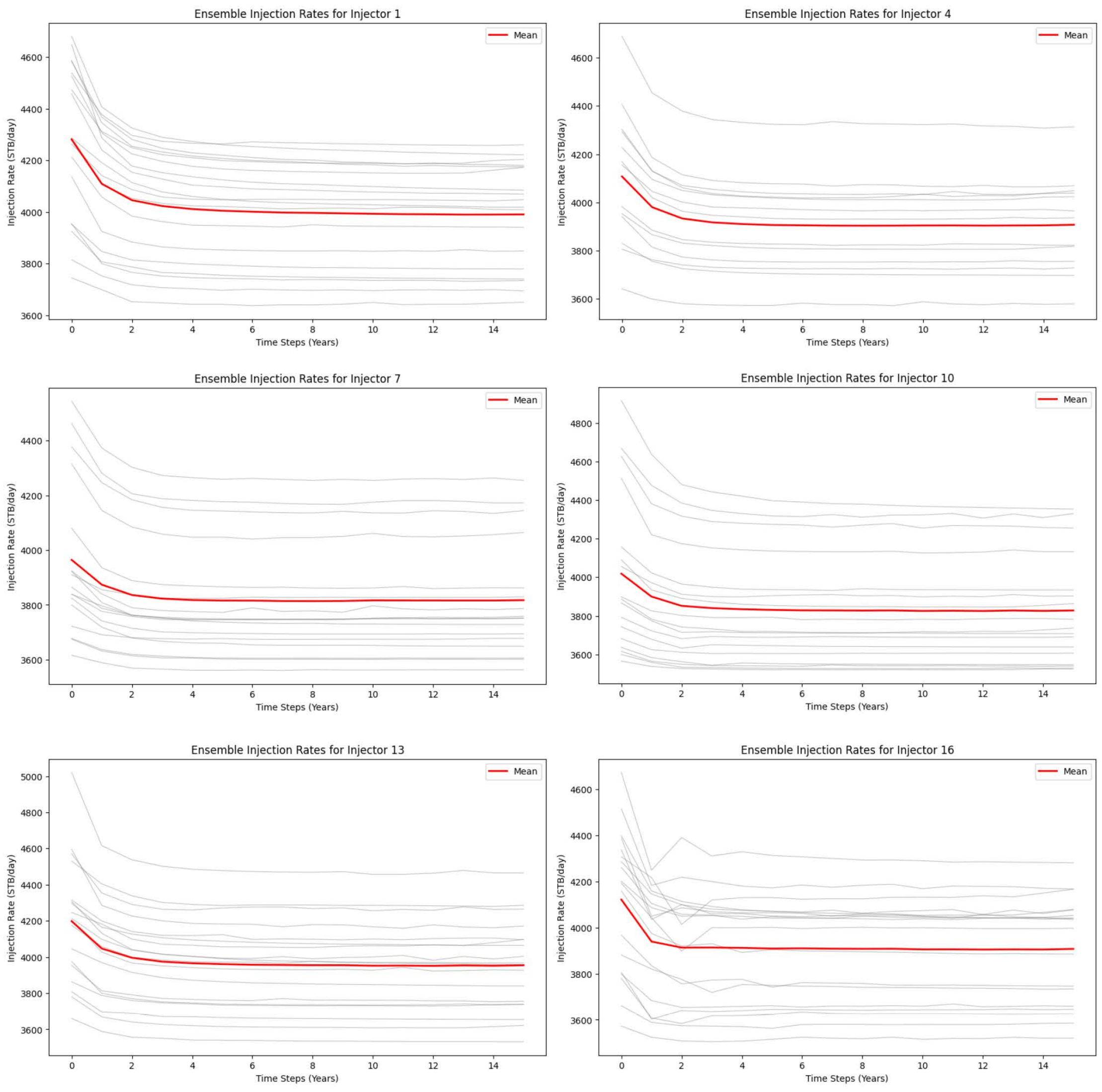


Figure 4—Injection rate of time for different injection wells in an ensemble of 20 cases.

Workflow Implementation

Working with a 3D grid, we use the 3D FNO implementation in PyTorch (Paszke A 2019). The model takes as input 11 channels, namely pressure, saturation, permeabilities in x, y, and z, pore volume, grid x, y, z co-ordinates, well injection rates at a given cell and well bottom hole pressure for producer wells at the given cell. The well controls are assigned using log scale of the controls at the grid locations of the corresponding well completions. The log values are taken to increase performance by differentiating between high values as most of the grid will contain 0 values for the well control channels, which makes normalization less effective directly. The model aims to predict the next timestep pressure values using these inputs and then auto-regressively predicts future pressure values by taking as input the pressure prediction from the previous inference. The loss function used in this model is the mean square error of the output with the true value. To accurately predict long rollouts in an auto-regressive manner, a rollout loss is implemented where the loss from the model prediction is aggregated over several timesteps *r* before it is back-propagated, see Equation (11). Also, since in the FNO dataset, we interpolate the simulation grid onto a bounding box, parts of the grid will have inactive cells. We only calculate the loss using active cells and optimize the parameters using these values. The optimal hyperparameters found during training using *ray-tune.* Note that in this specific reservoir configuration, we do not see much variation in the pressure field and are able to set a high rollout period during training without seeing a drop in performance.

$$\mathcal{L} = \frac{1}{r}\sum_{i=t}^{t+r}\left(\hat{\boldsymbol{y}}_{\boldsymbol{i}} - \boldsymbol{y}_{\boldsymbol{i}}\right)^2 \tag{11}$$

The GNN model is implemented by first constructing a graph which takes as nodes the active cells in the grid and the edges as the cell connections between the active cells. We apply a heuristic approach to attribute edge strengths to the edges to provide the graph with additional signal as to the rate of propagation of information. To do this, at each edge we calculate the harmonic mean of the permeabilities in each direction and weight them based on the distance in each direction and normalize by sum of the distances. This modifies the GCN II equation to Equation (12). We only use residual connections and drop identity maps in our implementation. To increase expressivity of the model and to allow for longer range interactions to be captured external of the GCN II framework, we construct an encoder and decoder before and after the GCN II layers respectively to map the input features to a hidden dimension and from the hidden dimension to the output dimension. These two structures are feedforward neural networks with a fixed depth. The full architecture is given in Figure 5. The GNN model is trained to predict the saturation values by taken the same features as the FNO model with the exception that the pressure is that of one timestep ahead of the saturation values. This is because, in the coupled regime, we solve for the pressure first and use that as input to predict saturations as is detail further below. We calculate the loss as the mean squared error loss at each node in the graph and perform a rollout aggregation like the FNO. The optimal hyperparameters during training are calculated with *raytune* (Richard Liaw 2018). Note that here, since there is a lot more change in the saturation field compared to the pressure, we have had to reduce the rollout period.

$$\boldsymbol{h}_u^{(\boldsymbol{l}+\boldsymbol{1})} = \sigma\left(\alpha \boldsymbol{h}_u^{(\boldsymbol{0})} + (1-\alpha)\boldsymbol{W}^{(\boldsymbol{l})}\sum_{\boldsymbol{v}\in N_{\boldsymbol{u}}\cup\{\boldsymbol{u}\}} d\left(\frac{\boldsymbol{2k_u k_v}}{\boldsymbol{k_u}+\boldsymbol{k_v}}\right)\boldsymbol{h}_{\boldsymbol{v}}^{(\boldsymbol{l})}\right) \tag{12}$$

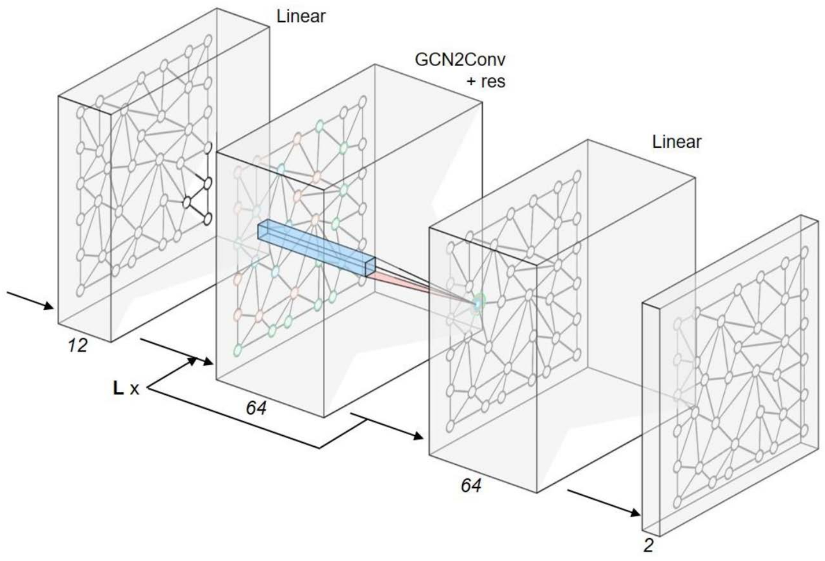


Figure 5—Full model architecture for the modified Graph Neural Network with encoder and decoder layers.

The above is the final GCN equation which we implement. Here k_u and k_v are the permeabilities of the two nodes of the connected edge and d is a function that weights the harmonic means based on distances in each direction.

The well properties prediction model is a feedforward neural network which takes as input the pressure and saturation values at each of the well to cell connections in the grid for a given well and the well control for that grid and predicts the production rates of water and oil for producer wells and the well bottom hole pressure for the injector wells. This is two separate networks for injector and producer wells. The model is trained over all the wells in the reservoir and use mean square error loss to train the model. The well output network is primarily used to test the effectiveness of the FNO-GNN coupling and hence chosen to have a simple architecture so as to not potentially add additional large sources of error in the workflow.

Once these three models are trained, we construct the full coupled workflow for inference. First, at the initial state ($t = 0$), we performance inference with the FNO to get the pressures at $t = 1$. These pressure values are taken as input along with saturation values at $t = 0$ to the GNN model to get predictions for saturations at $t = 1$. Using the pressures and saturations at the first timestep and well controls at $t = 1$, we perform inference with the well models to get the outputs for all wells in the reservoir. (Note that well controls at $t = 0$ will all be 0). Then, using the output from the GNN and the previous output from the FNO, we perform inference with the FNO to get pressures at $t = 2$, which are then fed into the GNN to get saturations at $t = 2$. These values are then fed to the well models again to get the well outputs at $t = 2$. This procedure is performed iteratively T times until we arrive at the desired end of the time stepping.

Algorithm 1—Inference loop for coupled workflow

```
Input: $p^0, s^0, q^{1:T}, z, FNO, GNN, WM$. Here $p^0, s^0, z$ are the initial pressures and saturations and static properties of the reservoir. $q^{1:T}$ are the well controls from the first timestep to the end of the simulation. FNO, GNN and WM are the respective trained models.
for $t=0$ to $T-1$ do
    $p^{t+1} \leftarrow FNO(p^t, s^t, z)$
    $s^{t+1} \leftarrow GNN(p^{t+1}, s^t, z)$
    $w^{t+1} \leftarrow WM(q^{t+1}, p^{t+1}, s^{t+1}, z)$
Output: $p^{1:T}, s^{1:T}, w^{1:T}$. Where $w^{1:T}$ corresponds to the well production rates.
```

Results

Before we present the results for the full 3D coupled workflow, we present results to show that the GNN by itself can predict pressures and saturations on a 2D case for varying well injection rates for a CO_2 injection ensemble. The 2D case consists of a 37×50 rectangular grid simulated over 1000 years, ~12 years of 143 monthly timesteps and the remainder being post-injection steps, with one well at the grid cell (31, 0). The post-injection phase timesteps increase progressively from 1, 10, 100 to eventually 500 years to model long-term containment behavior. The static properties for a single case are given in Figure 6. Here, since we only have one well, instead of passing the well controls as a channel in a single cell, we pass them as a multiplier to the edge attributes to increase the expressivity of the model. This is not as useful in the 3D case as we will have multiple well connections sharing edges. We use the same training procedure as the 3D GNN model and the same loss function but in inference we simply feed the predictions of the previous time as input into the next time in an auto-regressive manner to predict the rollout values.

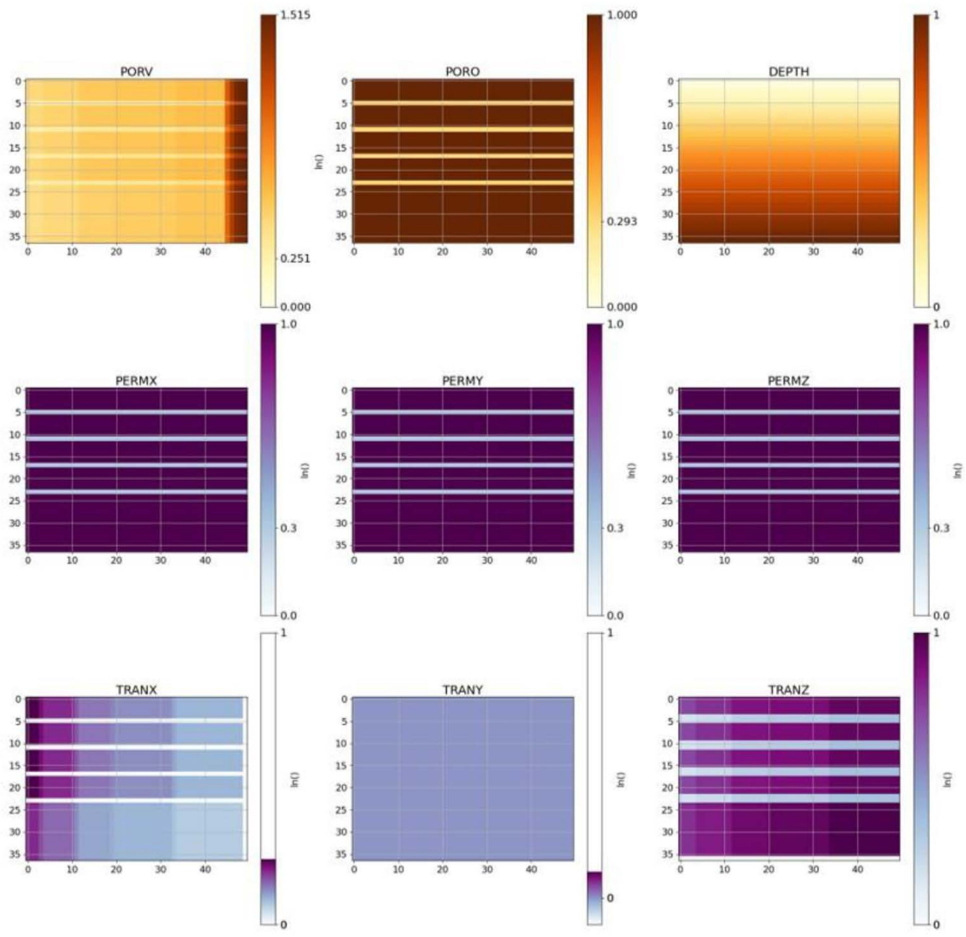
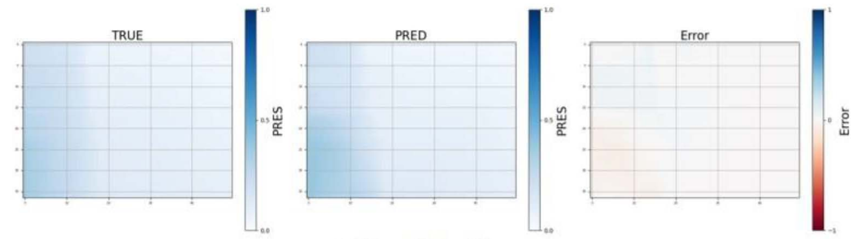
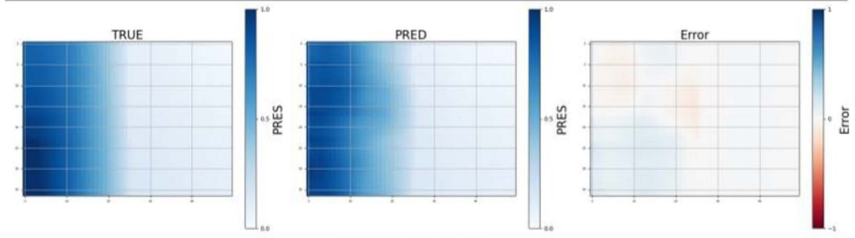


Figure 6—Static properties of a general 2D sample from the variable injection ensemble (constant across the ensemble). It shows the general structure with horizontal shale layers found in every 2D sample. PORV increases steadily in the x-direction. DEPTH increases in the increasing z-direction. PORO influences the input property PORV. PERMY is preconfigured nonzero, but is not encoded in any edge weight. Transmissibility depends on the cell dimension in its specific direction, making TRANY zero in the xz-dimensional grid. TRANX/TRANZ are influenced by PERMX/PERMZ, PORO and PORV. The variable shale ensemble is identical in the non-shale surroundings, but has varying PORO and randomly decaying PERMX/PERMZ in the shale layers.

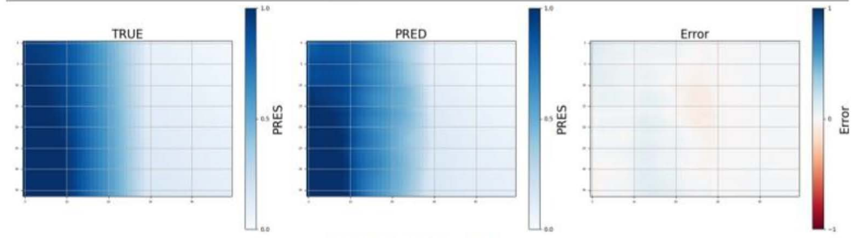
The results for the pressure and saturation predictions for three different cases with low, medium and high injection rates respectively, at timestep 80 are given in Figures 7 and 8. From the results, we see that the GNN architecture can auto-regressively predict both the pressures and saturations accurately for all three cases. This is mainly attributed to the edge multiplier added from the injection rates which provides information for the rate of flow. We see that in the saturation predictions, the front has passed further for the higher injection rate case as is to be expected and in the pressure predictions, the magnitude of the pressures is significantly higher for the high injection rate case. This shows that the model can effectively learn propagation rates for different scenarios in this 2D CO_2 injection ensemble.



(a) Case 25, *t* = 80

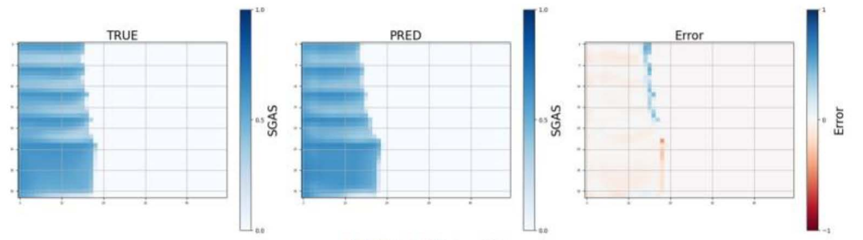


(b) Case 55, *t* = 80

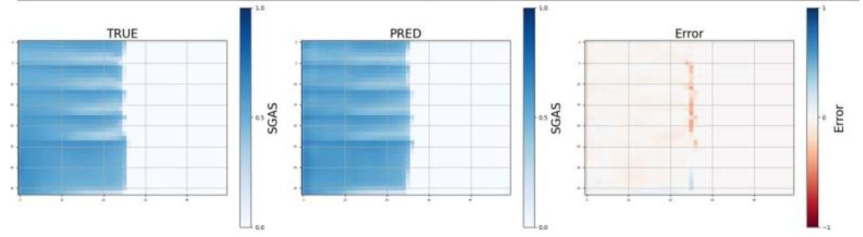


(c) Case 85, *t* = 80

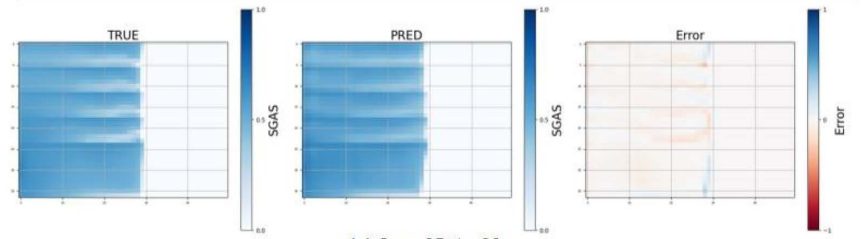
Figure 7—Pressure (PRES) simulation, roll-out prediction and error at t = 80 for the variable injection ensemble. a) Low injection rate, b) medium injection rate, c) high injection rate



(a) Case 25, *t* = 80



(b) Case 55, *t* = 80



(c) Case 85, *t* = 80

Figure 8—Gas saturation (SGAS) simulation, roll-out prediction and error at t = 80 for the variable injection ensemble. a) Low injection rate, b) medium injection rate, c) high injection rate

Pressure Results

Now, we present the results for the full 3D case. The results from the FNO on pressure predictions are given in Figure 9. Here we present the results for an unseen case at target times of 1 year, 4 years, and 7 years for a single layer. We observe that at all the times, the shape of the pressure field matches almost identically between the predicted and true values, however this is made easier because we do not observe a significant change in the pressures over time in the reservoir anyway. This also allowed us to increase the rollout loss period to 10 timesteps to more accurately auto-regressively predict the future values without incurring a significant compounding loss. We notice that the pressure field interaction between the injector and producer wells are also captured accurately. The highest regions of error are found near the injection wells and to the edge of the reservoir. It is expected that we observe the errors near the injection wells as this is where pressure changes most with varying injection rates and the model needs to adapt to such changes using the well control encoding. In the specific case presented, the error near the bottom right is due to the fact that an injector well in that region was shut off during the simulation but the well control encoding via the additional channel was not able to influence the shut off in the machine learning prediction. This indicates that a more rigorous technique needs to be employed to pass information on well controls to the model.

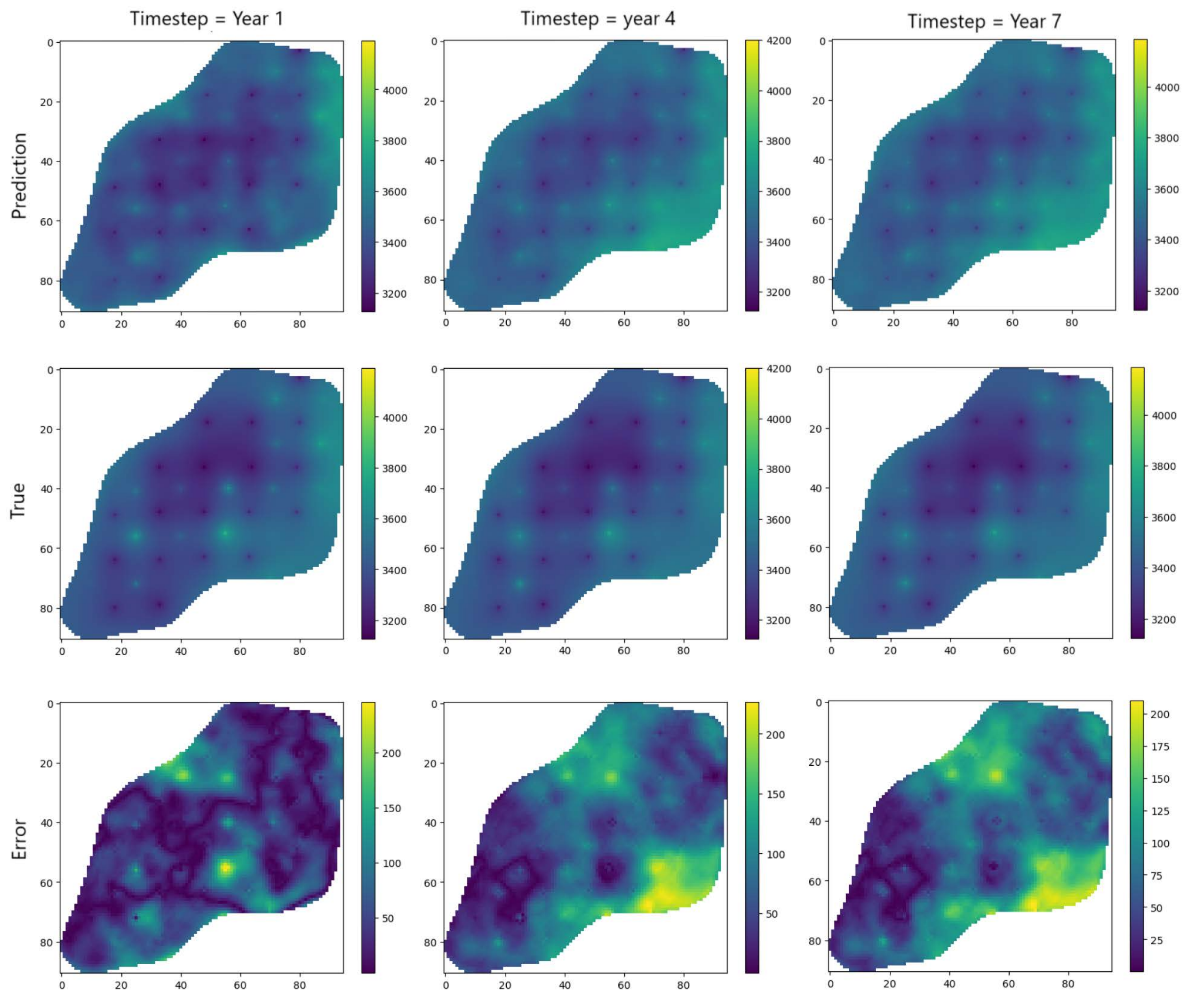


Figure 9—Pressure predictions for a layer in a case over 8 timesteps. (Here timestep refers to the target time in the simulation at which the results are presented).

Saturation Results

In Figure 10, we present the results of an unseen case at a specific layer. We note that the model is able to capture the location of wells and propagate saturations outward accurately. It is also able to capture the shape of the saturation in most cases. Where it underperforms, it is likely due to the fact that we are working with a 3D model and slight differences in the saturation field at a different layer can compound and drastically affect the results in a different layer. We observe that the majority of the errors are on the saturation fronts and areas in between are predicted almost perfectly, which would be much harder for the FNO compared to local aggregation in the GNN. We note a similar issue to the FNO where we see a well which is shut off in the top right corner is not captured to be shut in the machine learning model. This points once again to the lack of effectiveness of the well encoding in the machine learning model.

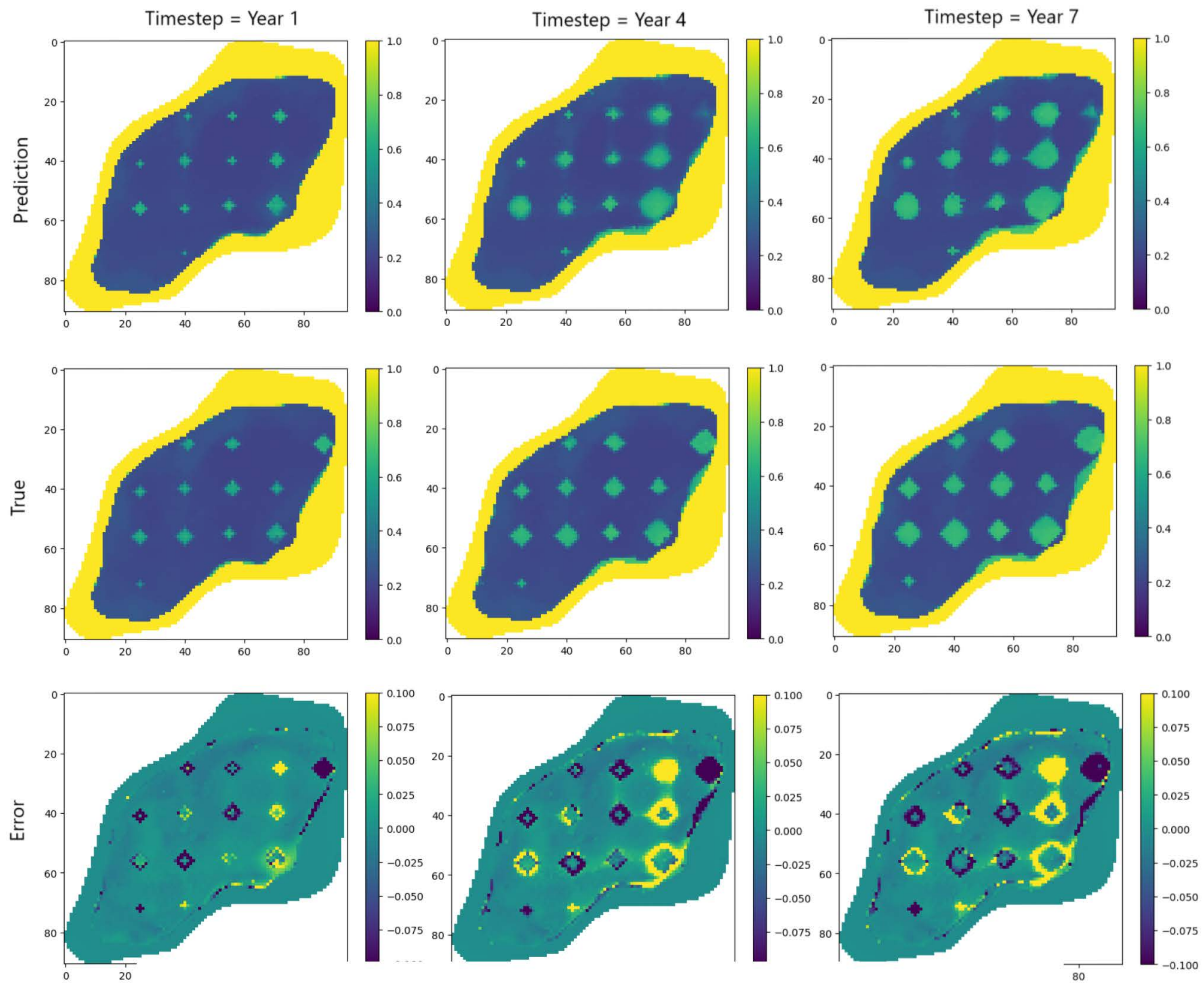


Figure 10—Saturation predictions for a layer in a case over 8 timesteps. (Here timestep refers to the target time in the simulation at which the results are presented).

Well Output Results

Figure 11 shows the well outputs from the coupled models. We show that even with the saturation values showing errors towards the fronts, we are able to accurately predict well outputs within a 10% error range for almost all timesteps in all the producer wells. This is due to the fact that the well network is only dependent on the pressures and saturations of the cells at the well to cell connection points which tend to be accurately predicted by the GNN-FNO coupled model. We note that when a well is shut off, the model accurately captures this and predicts 0 output as the well controls are passed into the well model as input. This is different from in the FNO and GNN models which are not able to accurately predict well shut off.

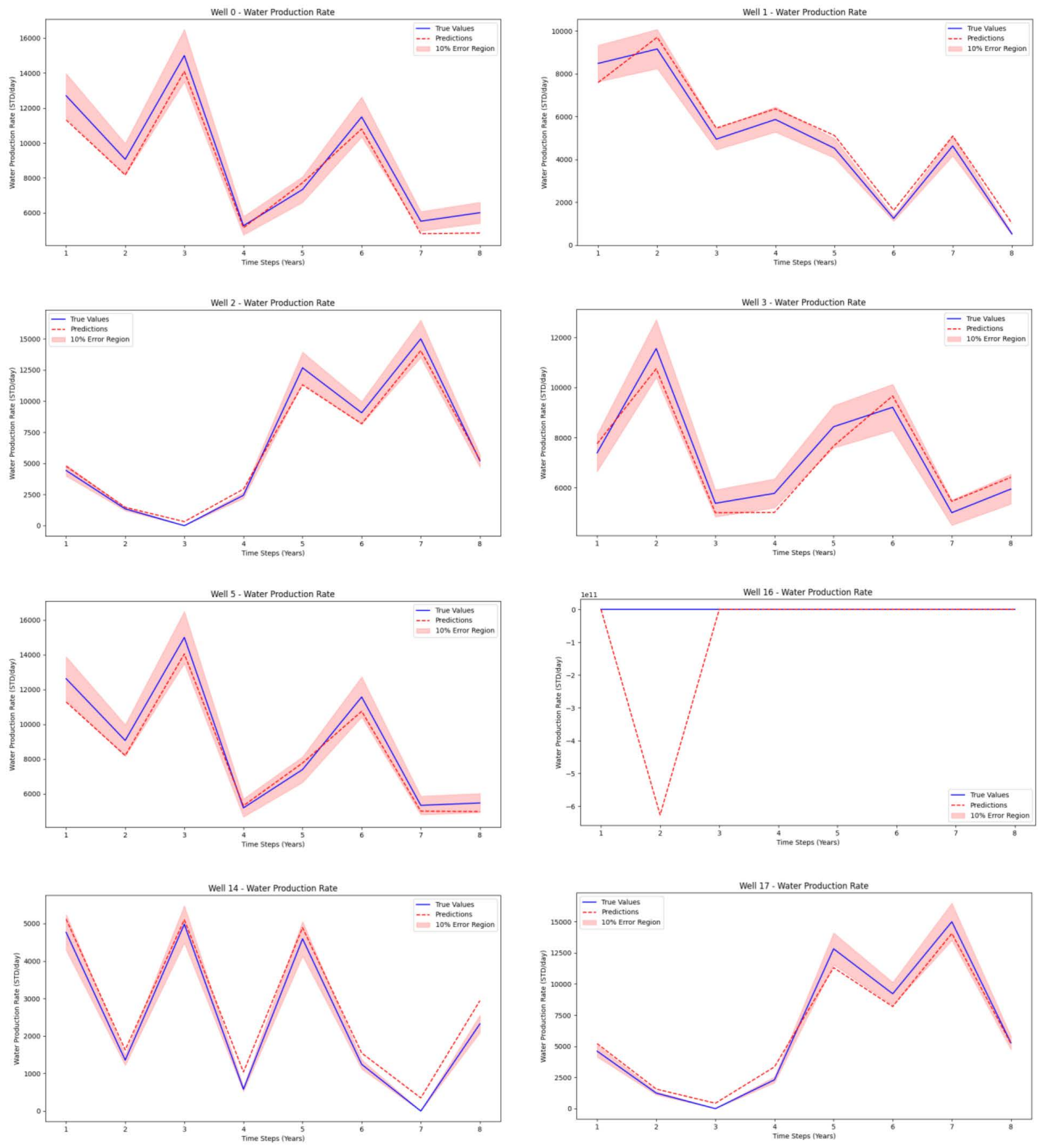


Figure 11—Well production rate predictions in a case over 8 timesteps for a selection of 8 wells. (Here timestep refers to the target time in the simulation at which the results are presented).

Conclusion

In this work, we have proposed a novel coupled FNO-GNN architecture to auto-regressively predict pressures, saturations and well production rates of a complex 3D reservoir ensemble under varying well controls. We show that this model can accurately predict the pressures and saturations of unseen cases several years into the future where only the initial state and well controls given. It is also able to predict well production rates within a 10% error rate for most wells in the ensemble. In future work, we aim work on generalizing this work by also varying the permeability fields and well locations and introduce specific mechanisms which allow us to handle this alongside varied well controls.

References

Atadeger, A., Sheth, S., Vera, G., Banerjee, R., Onur, M. 2022. "Deep learning-based proxy models to simulate subsurface flow of three-dimensional reservoir systems." *Eur Assoc Geosci Eng*.

Bhattacharya, Kaushik, Bamdad Hosseini, Nikola B. Kovachki, and Andrew M. Stuart. 2021. "Model reduction and neural networks for parametric PDEs." *The SMAI Journal of computational mathematics* Volume **7** pp. 121–157.

Coutinho EJR, Dall'Aqua, M., Gildin, E. 2021. "Physics-aware deep-learning-based proxy reservoir simulation model equipped with state and well output prediction." *Front Appl Math Stat*.

He, K., Zhang, X., Ren, S., Sun, J. 2016. "Deep residual learning for image recognition." IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas. 770–778.

Huang, Hu, Bin Gong, and Wenyue Sun. 2023. "A Deep-Learning-Based Graph Neural Network-Long-Short-Term Memory Model for Reservoir Simulation and Optimization With Varying Well Controls." *SPE Journal*.

J. Costa Gomes, S. Geiger, D. Arnold. 2023. "The Design of an Open-Source Carbonate Reservoir Model." *European Association of Geoscientists & Engineers* (European Association of Geoscientists & Engineers) 1–5.

Jiang, Z., Zhu, M., Lu, L. 2024. "Fourier-MIONet: Fourier-enhanced multiple-input neural operators for multiphase modeling of geological carbon sequestration." *Reliability Engineering & System Safety*.

Jin, Z.L., Liu, Y., Durlofsky, L.J. 2020. "Deep-learning-based surrogate model for reservoir simulation with time-varying well controls." *J Pet Sci Eng*.

Li, Z., Kovachki, N.B., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., et al 2021. "Fourier neural operator for parametric partial differential equations." *International Conference on Learning Representations*.

Michael M. Bronstein, Joan Bruna, Taco Cohen, Petar Veličković. 2021. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges. arXiv.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, Anima Anandkumar. 2023. "Neural Operator: Learning Maps Between Function Spaces With Applications to PDEs." *Journal of Machine Learning Research* 1–97.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., et al 2019. "PyTorch: An Imperative Style, High-Performance Deep Learning Library." *Advances in Neural Information Processing Systems* **32**. 8024–8035.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, Yoshua Bengio. 2018. "Graph Attention Networks." International Conference on Learning Representations.

Qimai Li, Zhichao Han, Xiao-Ming Wu. 2018. "Deeper insights into graph convolutional networks for semi-supervised learning." Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence. New Orleans, Louisiana, USA: AAAI Press.

Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E. Gonzalez, Ion Stoica. 2018. "Tune: A Research Platform for Distributed Model Selection and Training." ICML AutoML Workshop.

Ronneberger, O., Fischer, P., Brox, T. 2015. "U-Net: Convolutional networks for biomedical image segmentation." Medical Image Computing and Computer-Assisted Intervention - MICCAI. Springer.

Sasal, L., D. Busby, and A. Hadid. 2024. "A Graph Neural Network-Based Approach for Complex Reservoirs Simulation Surrogate Modelling." ECMOR 2024. European Association of Geoscientists & Engineers.

Sathujoda, S., Sheth, S.M. 2023. "Physics-informed localized learning for advection-diffusion-reaction systems." *ICML Workshop on New Frontiers in Learning, Control, and Dynamical Systems*.

Sathujoda S., Veeger, L., Sheth, S., Jönsthövel, T., Yorke-Smith, N. 2024. "Modeling Complex Fluid Flow in Porous Media Using Graph Neural Networks." *European Conference on the Mathematics of Geological Reservoir*. Oslo: European Association of Geoscientists & Engineers. 1–20.

Surya Sathujoda, Yuan Wang, Kanishk Gandhi. 2023. "Exciton-Polariton Condensates: A Fourier Neural Operator Approach." 37th Conference on Neural Information Processing Systems. AI for Science Workshop.

Thomas N. Kipf, Max Welling. 2017. "Semi-Supervised Classification with Graph Convolutional Networks." International Conference on Learning Representations.

Watter, M., Springenberg, J.T., Boedecker, J., Riedmiller, M. 2015. "Embed to control: A locally linear latent dynamics model for control from raw images." Proceedings of the *28th International Conference on Neural Information Processing Systems* - Volume **2**.

Wen, G., Li, Z., Azizzadenesheli, K., Anandkumar, A., Benson, S.M. 2022. "U-FNO—An enhanced Fourier neural operator-based deep-learning model for multiphase flow." *Advances in Water Resources*.

Wen, G., Li, Z., Long, Q., Azizzadenesheli, K., Anandkumar, A., Benson, S.M. 2023. "Real-time high-resolution CO 2 geological storage prediction using nested Fourier neural operators." *Energy & Environmental Science* 1732–1741.

Zhang, K., Zuo, Y., Zhao, H., Ma, X., Gu, J., Wang, J., Yang, Y., Yao, C., Yao, J. 2022. "Fourier neural operator for solving subsurface oil/water two-phase flow partial differential equation." *Spe Journal*.

Zhu, Y., Zabaras, N. 2018. "Bayesian deep convolutional encoder-decoder networks for surrogate modeling and uncertainty quantification." *J Comput Phys*.